AFRL-RI-RS-TR-2008-279
In-House Technical Report
October 2008

DISTRIBUTED EPISODIC EXPLORATORY PLANNING (DEEP)

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

Qualified requestors may obtain copies of this report from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2008-279 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/ /s/

JOSEPH A. CAROZZONI
Work Unit Manager

JAMES W. CUSACK, Chief
Information Systems Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
OCT 08	Interim	July 06 – June 08

4. TITLE AND SUBTITLE

DISTRIBUTED EPISODIC EXPLORATORY PLANNING (DEEP)

5a. CONTRACT NUMBER
In-House

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
62702F

6. AUTHOR(S)

Joseph A. Carozzoni, James H. Lawton, Chad DeStefano, Anthony J. Ford, Jeffrey W. Hudack, Kurt K. Lachevet, and Gennady R. Staskevich

5d. PROJECT NUMBER
558S

5e. TASK NUMBER
IH

5f. WORK UNIT NUMBER
DP

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/Information Directorate
Rome Site/RISB
525 Brooks Rd.
Rome NY 13441-4505

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/Information Directorate
Rome Site
26 Electronic Parkway
Rome, NY 13441

10. SPONSOR/MONITOR'S ACRONYM(S)
N/A

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TR-2008-279

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED*.PA# 08-0584*

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This report describes an overview and the progress to date of the Distributed Episodic Exploratory Planning (DEEP) project. DEEP is a mixed-initiative decision support system that utilizes past experiences to suggest courses of action for new situations. It has been designed as a distributed multi-agent system, using agents to maintain and exploit the experiences of individual commanders as well as to transform suggested past plans into potential solutions for new problems. The system is mixed-initiative in the sense that a commander, through his or her agent, can view and modify the contents of the shared repository as needed. The agents interact through a common knowledge repository, represented by a blackboard in the initial architecture. The blackboard architecture is well suited for dealing with ill-defined, complex situations such as warfare.

15. SUBJECT TERMS
Distributed Command and Control, Analogical Reasoning, Network Centric Planning, Intelligent Agents

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Joseph A. Carozzoni
U	U	U	UU	44	19b. TELEPHONE NUMBER (*Include area code*) N/A

Abstract

This report describes an overview and the progress to date of the Distributed Episodic Exploratory Planning (DEEP) project. DEEP is a mixed-initiative decision support system that utilizes past experiences to suggest courses of action for new situations. It has been designed as a distributed multi-agent system, using agents to maintain and exploit the experiences of individual commanders as well as to transform suggested past plans into potential solutions for new problems. The system is mixed-initiative in the sense that a commander, through his or her agent, can view and modify the contents of the shared repository as needed. The agents interact through a common knowledge repository, represented by a blackboard in the initial architecture. The blackboard architecture is well suited for dealing with ill-defined, complex situations such as warfare.

Contents

List of Figures

1 Executive Summary

This report describes an overview and the progress to date of the Distributed Episodic Exploratory Planning (DEEP) project. DEEP is a mixed-initiative decision support system that utilizes past experiences to suggest courses of action for new situations. It has been designed as a distributed multi-agent system, using agents to maintain and exploit the experiences of individual commanders as well as to transform suggested past plans into potential solutions for new problems. The system is mixed-initiative in the sense that a commander, through his or her agent, can view and modify the contents of the shared repository as needed. The agents interact through a common knowledge repository, represented by a blackboard in the initial architecture. The blackboard architecture is well suited for dealing with ill-defined, complex situations such as warfare.

The DEEP project was initiated in response to the need to support the key tenets of Network Centric Operations (NCO), namely information sharing, shared situational awareness, and knowledge of commander's intent. To that end, the project's long-term goal is to develop, in-house, a prototype system for distributed, mixed-initiative planning that improves decision-making by applying analogical reasoning over an experience base.

The core of this report documents the successful completion of the project's short-term objective: the development of a "research platform" to further support more aggressive research in the areas of distributed Command and Control (C2) and analogical reasoning, and how to apply the technology to advance the state of C2. This research platform implements DEEP's high-level system-of-systems architecture comprised of the following systems:

- Distributed Blackboard for multi-agent, non-deterministic, opportunistic reasoning
- Case-Based Reasoning to capture experiences (successes and/or failures)
- Episodic Memory for powerful analogical reasoning
- Multi-Agent System for mixed initiative planning
- ARPI Core Plan Representation for human-to-machine common dialog
- Constructive Simulation for exploration of possible future states

Our research platform implementation will serve the DEEP project both as a concept demonstration of how experience-based, distributed mixed-initiative planning can be accomplished in a network-centric environment, as well as an environment for conducting research on the individual systems needed to support this NCO vision. This platform enables research in the areas of semantic extensions to CPR, our plan representation; the use of *robust coherence* to utilize the experiences from several agents to solve a problem; the leveraging of distributed database technology to provide persistent storage for plans and planning information; a speech-act-based messaging formalism for consistent communications among the distributed system; and the use of advanced simulation platforms to improve the fidelity of plan execution and analysis.

2 Introduction

DEEP is a mixed-initiative decision support system that utilizes past experiences to suggest courses of action for new situations. It has been designed as a distributed multi-agent system, using agents to maintain and exploit the experiences of individual commanders as well as to transform suggested past plans into potential solutions for new problems. The system is mixed-initiative in the sense that a commander, through his or her agent, can view and modify the contents of the shared repository as needed. The agents interact through a common knowledge repository, represented by a blackboard in the initial architecture. The blackboard architecture is well suited for dealing with ill-defined, complex situations such as warfare.

2.1 Problem Statement

The U.S. and other highly industrialized nations have developed military capabilities that excel in conventional force-on-force warfare, especially where tactics are well developed and known. However, modern adversaries have devised the strategy of not going "head-to-head" with these capabilities and instead combat modern conventional forces with unconventional tactics. One example of the result of a weapon system being vastly superior is the case of the air superiority fighter which modern adversaries totally avoid putting themselves in a position to contest them.

To meet these future challenges, U.S. forces are in the midst of a "transformation" to not only support traditional high-tempo, large force-on-force engagements, but also smaller-scale conflicts characterized by insurgency tactics and time-sensitive targets of opportunity. This transformation requires a vastly new C2 process that can adapt to any level of conflict, provides a full-spectrum joint warfighting capability, and can rapidly handle any level of complexity and uncertainty.

To meet future challenges, the United States. Air Force (USAF) is moving towards a model of continuous air operations not bounded by the traditional 24-hour Air Tasking Order (ATO) cycle. Meeting these objectives will require a highly synchronized, distributed planning and replanning capability. As a potential way ahead, AF/A5 (Plans) released in May 2006 a revolutionary vision paper titled "C2 Enabling Concepts" (Braun, 2006)depicting what a potential future C2 environment could be. Four key concepts emerged as being critical to the success of a future Air Operations Center (AOC):

- Distributed/Reachback planning
  - "Today's constantly engaged AOCs no longer focus on shifting from one relatively rare major combat operation to the next. Their challenge is day-to-day, steady state C2 of these continual lower-end contingencies."

- Redundant/Backup planning

    - "AOCs can be geographically separated but electronically plugged into the battlespace so that they function as if their members were collocated."
- Continuous planning
    - "The supporting and supported AOCs will maximize use of distributed network capabilities to ensure information is backed up and the supporting AOC is prepared to assume operations should the engaged AOC encounter a catastrophic event that makes operations unsupportable."
- Flexible, scalable, tailorable C2
    - "..rapidly adapt to the level of conflict by connecting with worldwide capabilities, including joint and coalition forces."

Experience with recent operations also reveals that the C2 process must transition from a process of *observation and reaction* to one of *prediction and preemption*. To achieve this, we will need to go beyond the focus of military operations, and instead address the entire spectrum of Political, Military, Economics, Social, Infrastructure, and Information (PMESII) (Alberts & Hayes, 2007).

To that end, the focus of the research reported here is on developing a C2 environment that supports the vision of *Network Centric Operations* (NCO) (Alberts & Hayes, 2007). The tenets of NCO are:

- Information sharing
- Shared situational awareness
- Knowledge of commander's intent

2.2 DEEP Project Objective

In response to the need to support these key NCO tenets, the long-term goal of the Distributed Episodic Exploratory Planning (DEEP) project is to develop, in-house, a prototype system for distributed, mixed-initiative planning that improves decision-making by applying analogical reasoning over an experience base. The two key objectives of DEEP are:

- Provide a mixed-initiative planning environment where human expertise is captured and developed, then adapted and provided by a machine to augment human intuition and creativity.
- Support distributed planners in multiple cooperating command centers to conduct distributed and collaborative planning.

That is, the architecture of DEEP was explicitly designed to support the key tenets of NCO in a true distributed manner. Because DEEP is not based on any current C2 system, we are able to explore concepts such as combining planning and execution to support dynamic replanning, to examine machine-mediated self-synchronization of distributed planners, and to experiment with the impact of trust in an NCO environment (e.g., “Good ideas are more important than their source”).

Alberts and Hayes (2007) advocate bold new approaches beyond current organizational process, focusing on what is possible for NCO. High priority basic research topics recommended as areas to systematically explore are:

1. Taxonomy for planning and plans;
2. Quality metrics for planning and plans;
3. Factors that influence planning quality;
4. Factors that influence plan quality;
5. Impact of planning and plan quality on operations;
6. Methods and tools for planning; and
7. Plan visualization

This report describes our approach to achieving this vision of NCO and presents the progress to date on the development of the DEEP prototype, especially as it relates to these priorities.

3 Methods, Assumptions, and Procedures

In this chapter we present an overview of the DEEP architecture, a description of the systems that comprise the DEEP prototype, and a discussion of how these systems interact. In our development of this architecture, we have given careful consideration to Alberts and Hayes (2007) priorities (Section 2.2), noting that considerable attention has been given to the first research topic through the use of the Core Plan Representation (CPR) as the knowledge structure tying the system together.

3.1 DEEP Architecture Overview

DEEP is a system-of-systems architecture (Figure 1), comprised of the following systems:

- Distributed Blackboard for multi-agent, non-deterministic, opportunistic reasoning
- Case-Based Reasoning system to capture experiences (successes and/or failures)
- Episodic Memory for powerful analogical reasoning
- Multi-Agent System for mixed initiative planning
- ARPI Core Plan Representation for human-to-machine common dialog
- Constructive Simulation for exploration of possible future states

The DEEP architecture also includes a messaging system, various knowledge objects, a shared data storage system, along with a number of agents, all described later in this chapter. For convenience, we will describe the pieces in the architecture in the order in which they might be typically used. One should bear in mind, however, that in this type of mixed-initiative system, there will rarely be a clean path from the initial planning problem to the final solution.

Consider the system-of-systems architecture depicted in Figure 1. The starting point for entry into the system occurs when a commander describes a new mission using a *planning agent* (1). The planning agent allows for the commander to input information into the system which defines their *current objectives*. These objectives, along with other information, such as resources, locations, and time constraints, are collectively known as the *situation*. This situation is then placed on the shared blackboard (2). The blackboard would in turn notify all registered systems of the existence of the new situation. Using the given situation, the other planning agents, with their associated case bases and cased-based reasoning capabilities, would each search their case base for relevant past experiences (3). These results are then modified to fit the current situation (4) and are posted to the blackboard as *candidate plans* (5). Once the candidate plans are on the blackboard, they are adapted by specialized *adaptation agents* to further refine these plans to meet the current situation (6). These plans are now ready to be critiqued by the *critic agents*.

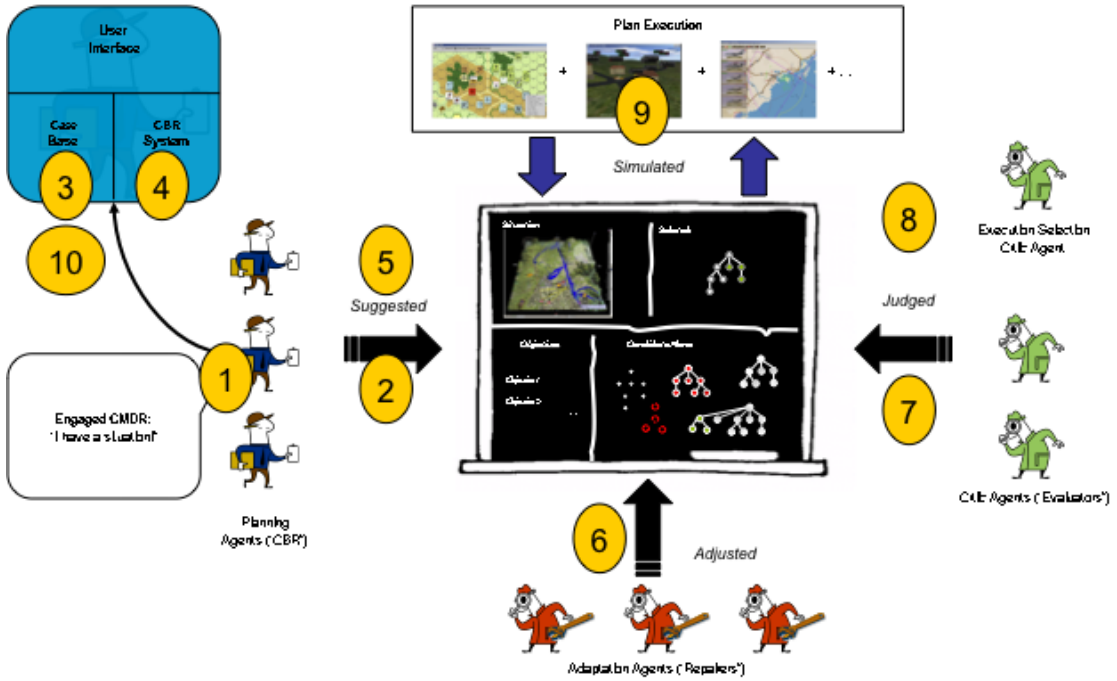

Figure 1 - DEEP Architecture

Critic agents concurrently scrutinize the candidate plans and score them based on their individual expertise (7). Once the plans are scored, the *execution selection critic* gathers the adapted plans along with their scores, determines their overall scores, and selects a number of top rated plans to be executed (8). The top rated plans are now executed (currently in a simulated environment) (9). Once a plan completes execution, the results are combined with the plan and assimilated back into the original planning agent's case base (10).

Although we have described this planning and execution as a single flow through the system, in reality few plans will execute without changes. The DEEP architecture supports the modification of currently executing plans through feedback of partial results of plan execution into the blackboard. This allows the plans to be run through the adaptation and critique processes as many times as needed.

In the remainder of this chapter we present the various systems of the DEEP architecture mentioned in the above flow example. We begin with an explanation of the system-of-system interaction facilities that underlie the DEEP prototype.

3.2 Systems Interaction

Being a network-centric application, there is a strong emphasis in DEEP on how knowledge is passed and how that knowledge is encapsulated. Before discussing the major structural systems of DEEP, we present a discussion on how the systems interact and on the knowledge objects that the systems process. The next two sections discuss the main type of knowledge object used in DEEP and in the DEEP messaging system.

3.2.1 Plan Representation

The various DEEP systems all use a common knowledge representation to facilitate their interactions. We know that the future of military planning is not just for the Air Force, but rather will involve participants from various agencies (both military and civilian), possibly planning at different levels of abstraction. Thus, DEEP was designed to

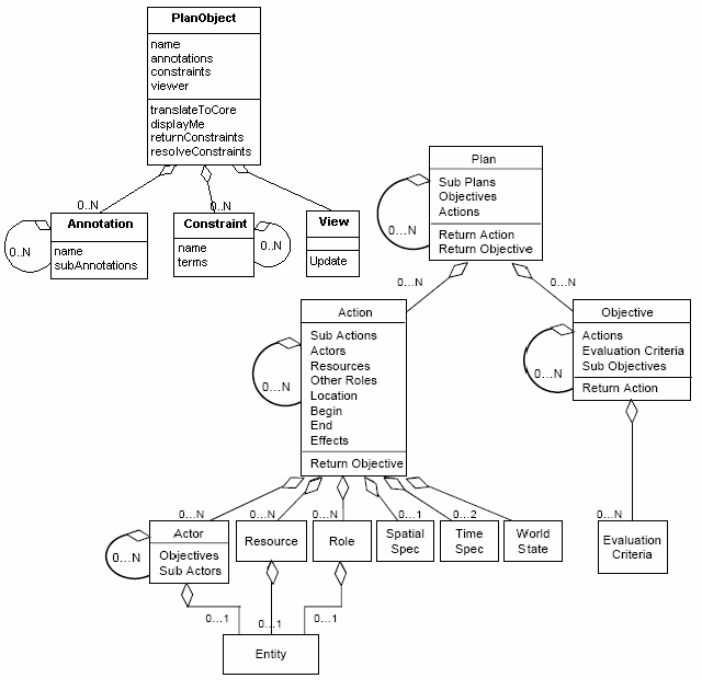

Figure 2 - Core Plan Representation

support plans for joint, coalition, and civilian operations as well handle plans at different abstraction levels (i.e., strategic, tactical, or operational). Planning for heterogeneous operations also means that the plan representation has to be able to consider the semantics of terms used in the plan, ensuring agreement among all participants. This is an ongoing research topic, discussed in detail in Section4.2. Finally, because DEEP is a mixed-initiative environment, the chosen plan representation must be easily machine-readable as well as presentable to a user.

The ARPA-Rome Laboratory Planning Initiative (ARPI) conducted research on several plan representations. The culmination of that effort was the Core Plan Representation (CPR), shown in Figure 2. CPR was selected for DEEP as best meeting the above criteria. CPR is an object-oriented structure that is agnostic to the planning abstraction level (i.e., strategic, tactical, or operational) (Pease A. , 1998). Its natural object oriented structure also lines up very well with the machine reasoning capability DEEP requires. We have adapted the original CPR structure (Figure 2) to meet the needs of DEEP as they have evolved over time.

In DEEP, CPR is used to represent individual experiences, or *cases*, which are composed of a plan, events, and one or more outcomes. The attributes of the plan are used by the cased-based reasoning system (Section 3.3.3) to determine the similarity of past cases with the current situation. Execution (currently through simulation) of the plan populates the events and outcome sections. DEEP-CPR (Figure 3) was extended from the base structure shown in Figure 2 to support a much deeper reasoning capability of plans.

3.2.2 System Messaging

CPR is the foundation for the DEEP architecture and used by all components, thus a formalized messaging model is required for the interactions within the systems. The systems that interact with one another include various types of agents along with the system blackboard. To accomplish this, a formalized messaging scheme based on inter-agent communication is required with a defined structure so that new systems are able to understand incoming messages as well as transmit their own. The DEEP architecture includes a formal messaging scheme to be used by the other systems.

In the current DEEP architecture, the communication protocol used is the publish-subscribe communication paradigm through the blackboard. At a high level, systems subscribe to the blackboard and are notified when new information is added. Because of the push to create a functional proof-of-concept architecture, a simple taxonomy is currently in place to determine notification and message types until a more formalized communications protocol is established. The blackboard mediates all messaging using its defined messaging scheme and connectivity medium. To prohibit the distributed planning aspect of DEEP deteriorating to "chat-room" type collaboration, an artificial barrier has been placed on human-to-human direct planning. Therefore, agents of any kind (human or software) do not communicate with each other directly, but instead use the blackboard as a hub of communication. For example, consider the mixed initiative scenario where a critic agent requires input from a user. To obtain this input, the critic agent would send a message through the blackboard to the appropriate interface agent; the reply would similarly be routed back through the blackboard.

As a more complex example, let us assume a simple setup of two planning agents (A and B), one blackboard, one critic agent, and one simulator. All systems are registered with the blackboard for communication. Planning Agent A becomes an *engaged* agent when its user inputs a new situation through the agent's user interface. Planning Agent A in turn places this new problem on the blackboard.

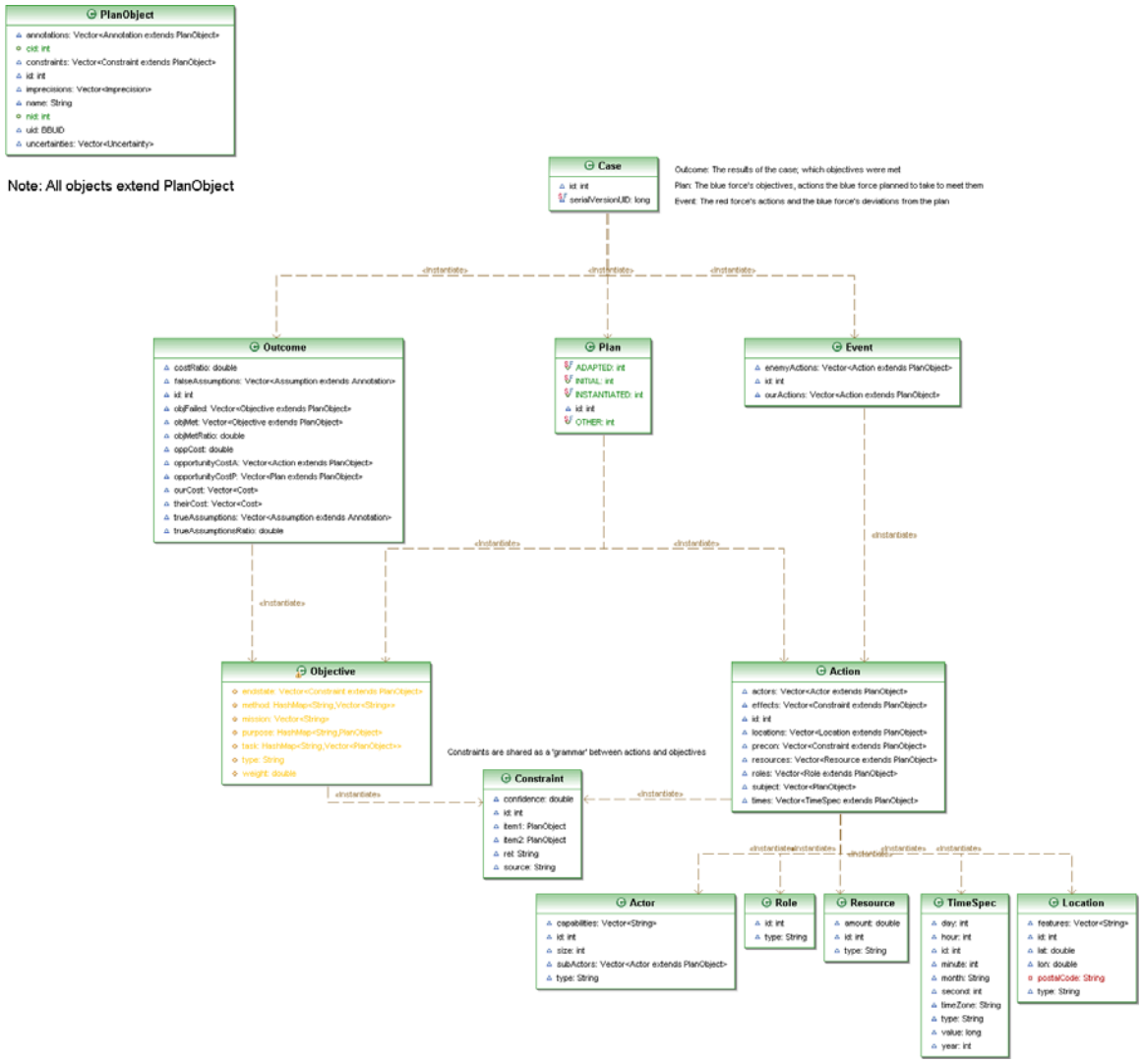

Figure 3 - DEEP-CPR

Once posted, the blackboard notifies registered systems with a message indicating the type of object (e.g., new problem situation) by broadcasting these messages. The notified systems have to decide if the message is relevant to them. This happens for all communication, so in our example when Planning Agent A posts a new situation, *all* registered system are notified, including critics who cannot do anything with a situation.

3.3 Architecture

We now explain the systems of the DEEP architecture that were mentioned in the previous section. These systems are the distributed blackboard, interface agents, and critic agents. The explanation will provide an understanding of how the pieces fit into the mixed-initiative distributed architecture of DEEP.

3.3.1 Blackboard – The Distributed Shared Knowledge Structure

As can be seen from the DEEP architecture in Figure 1, the various DEEP systems rely on a shared knowledge structure to act as a medium of communication and interaction. Also, in order to support the NCO vision discussed in Section 2.1, the DEEP architecture requires a mechanism that supports reach-back in a distributed system. A blackboard system was chosen to fulfill this need as it not only functions as a shared memory for the DEEP system, but as we discussed in Section 3.1, it provides other functionality as well.

A blackboard system is an opportunistic artificial intelligence application based on the blackboard architectural software engineering paradigm (Corkill, 1991).The blackboard system functions as a central knowledge store facilitating communication and interaction between the different software systems, including interface agents, critic agents, and simulation engines (explained later in this chapter). These interactions are made possible by the sharing and passing of objects.

To fully meet the requirements of the DEEP vision for distributed C2, a distributed blackboard system was required. Current commercial and open source blackboard system implementations are not distributed, so the paradigm needed to be extended from a monolithic to a distributed environment. The current DEEP blackboard, shown in Figure 4, was designed and implemented following this extended view using design patterns described in (Hughes & Hughes, 2003).

Traditionally, a blackboard consists of three discrete components: the blackboard knowledge structure which is a central repository for knowledge objects, the knowledge sources which are specialist software modules (agents in the DEEP software architecture) that provide specific expertise required by the system, and a control component which controls the flow of objects and problem-solving activity in the system (Corkill, 1991). Each of these systems will be described in detail in the following sections.

3.3.1.1 Core Knowledge Structure

The core knowledge structure (see Figure 4) of the blackboard is the global knowledge store labeled as “BB Data Structure” in the diagram. This knowledge structure holds the objects within the system and is accessible by all of the system’s knowledge sources. Because there could potentially be an extremely large number of objects placed on and contained within the blackboard at one time, blackboard data structures are conventionally divided in more than one way. These divisions are known as panes and layers, and could potentially contain further dimensions of separation (Corkill, 1991).

In the DEEP architecture, the core knowledge structure is defined to provide certain functionality. The knowledge store component of the blackboard has been abstracted out to allow for future revisions and extensions to how and where the knowledge is stored. This interface allows the option for the backend of the blackboard to be replaced with a database or other high performance data store.

Figure 4 - Distributed Blackboard Architecture

In DEEP's current state, the core data structure is a matrix of hash tables. These hash tables within hash tables allow panes and layers to be identified by providing a unique key and utilized to divide the blackboard up into more manageable sections. Now that there is a global repository for data, software system may start to utilize it to solve problems.

3.3.1.2 Knowledge Sources

By connecting to the blackboard, an application has the ability to become a *knowledge source* of the blackboard system. Knowledge sources are independent agents that provide specialized expertise that contributes to the solution of a problem. Figure 4 shows example knowledge sources from the DEEP architecture and an example of how a knowledge source could contribute external expertise to the blackboard. A key characteristic of knowledge sources in a blackboard system is that they require no knowledge of the other knowledge sources that are connected to the blackboard. They bring their specialized expertise to the system and do not rely on others to provide it. Each knowledge source is responsible for knowing when, what, and how it may contribute to the solution to the current problem on the blackboard (Corkill, 1991).

In the DEEP architecture, all systems must implement an interface provided by the blackboard in order to connect to it. The connection process includes connecting to the local blackboard proxy and registering with the blackboard for blackboard update event notifications (more on this in the control discussion in Section 3.3.1.3). So now there is a data store and knowledge sources are ready to place objects on it. The next section discusses the control mechanism of the blackboard to facilitate the problem solving which will take place on the blackboard.

3.3.1.3 Control

There are several control system paradigms that may be employed when designing a blackboard system. It may be very centralized to the blackboard, distributed among the blackboard and knowledge sources, or pushed out to the knowledge sources, requiring them to facilitate their own control of contributions to the problem (Corkill, 1991). The blackboard system developed for the DEEP architecture splits the control between the blackboard application and its knowledge sources. The control component on the actual blackboard application directs communication among the distributed blackboards, where the knowledge sources are held responsible for choosing whether or not they should interact with new or updated objects on the blackboard, or even taking initiative and placing a new object on the blackboard without waiting for a blackboard event notification.

The control component of the blackboard is also what enables multiple blackboards to remain synchronized and distributed. Because the control component manages all of the activity occurring within the blackboard system, it is able to control how information is distributed among the connected blackboards, as well as maintaining synchronization through the use of queues and messaging schemes. This is what allows the blackboard system to be viewed as a single logical blackboard, while physically there are multiple, synchronized replicated blackboards. When a new object is passed to the blackboard proxy by a knowledge source, it is passed through the control mechanism, which distributes it to all connected blackboard applications. After all the connected blackboard systems receive the new object, they are placed in their local data store waiting to be manipulated or retrieved.

In addition to the three traditional blackboard systems, the distributed blackboard designed for DEEP includes additional components that are necessary for a distributed blackboard and uniform communication between the knowledge sources connected to the distributed blackboard system.

3.3.1.4 Proxy

The proxy is an interface provided by the blackboard that allows a knowledge source to connect and interface with it. This proxy connection is established using a network socket. Originally, a blackboard application was designed to be running on each computer that contains one or more knowledge sources. However, because each knowledge source connects using a network socket, it may reside on a separate computer.

This proxy allows the interface to perform actions to the blackboard such as the posting and retrieval of objects. Other actions include the retrieval of an object by its unique identifier and the registration of new blackboard listeners. Similarly to the core data structure, the proxy interface could easily be extended to accommodate integration with other applications (new or existing) as needed.

3.3.1.5 Blackboard Objects

A well-defined common interaction language is also necessary for a successful blackboard system. To keep the distributed blackboard as flexible as possible, the blackboard provides a simple interface to the knowledge sources for objects to be placed on the blackboard. This interface forces objects placed on the blackboard to contain certain properties and functions so the blackboard can work with the object. The properties include the partition the object belongs to, a unique identifier (UID) for each object, and a timestamp. By implementing this interface, the object also becomes serializable, allowing it to be transmitted over a network socket.

3.3.1.6 Blackboard Utilities

One of the main blackboard utilities is the *Packet*. A packet in this context is utilized by the blackboard control system to send messages to other connected blackboards, and is what the knowledge sources receive when they get an update event from the blackboard. Depending on the packet type, it contains certain useful information, some containing blackboard objects.

Another blackboard utility is the Blackboard Unique Identifier (BBUID), which is a unique identifier across a network. This UID is required for all blackboard objects, system, and knowledge sources. There are also other convenience utilities such as a log writer and properties file parser.

The Distributed blackboard is an integral part of the DEEP architecture in that it provides the functionality and ability for the system to become distributed. Now that there is a distributed data structure, we can look at the DEEP knowledge sources and how they will utilize the blackboard.

3.3.2 DEEP Agents

The DEEP system uses two different types of agents. The first type is called an *interface* or *planning* agent and the second type is a *critic* agent.

Agents in the DEEP architecture extend and use the Java Agent DEvelopment (JADE) framework (Bellifemine, 2006). DEEP requires a distributed multi-agent system and a framework to help simplify the implementation of this system. JADE was chosen because it is fully implemented in Java, and supports these requirements.

3.3.3 Interface Agents

Interface agents, also called planning agents, serve a dual purpose. From the user perspective, these agents provide the interface into the DEEP system, enabling the user to input new situations and communicate with the DEEP system. These agents also provide the wrapper for the user's case base (representing his or her planning experiences) and incorporate a case-based reasoner that utilizes the case base for planning. That is, these agents are the user front end for the planner and allow for mixed-initiative interaction with the system.

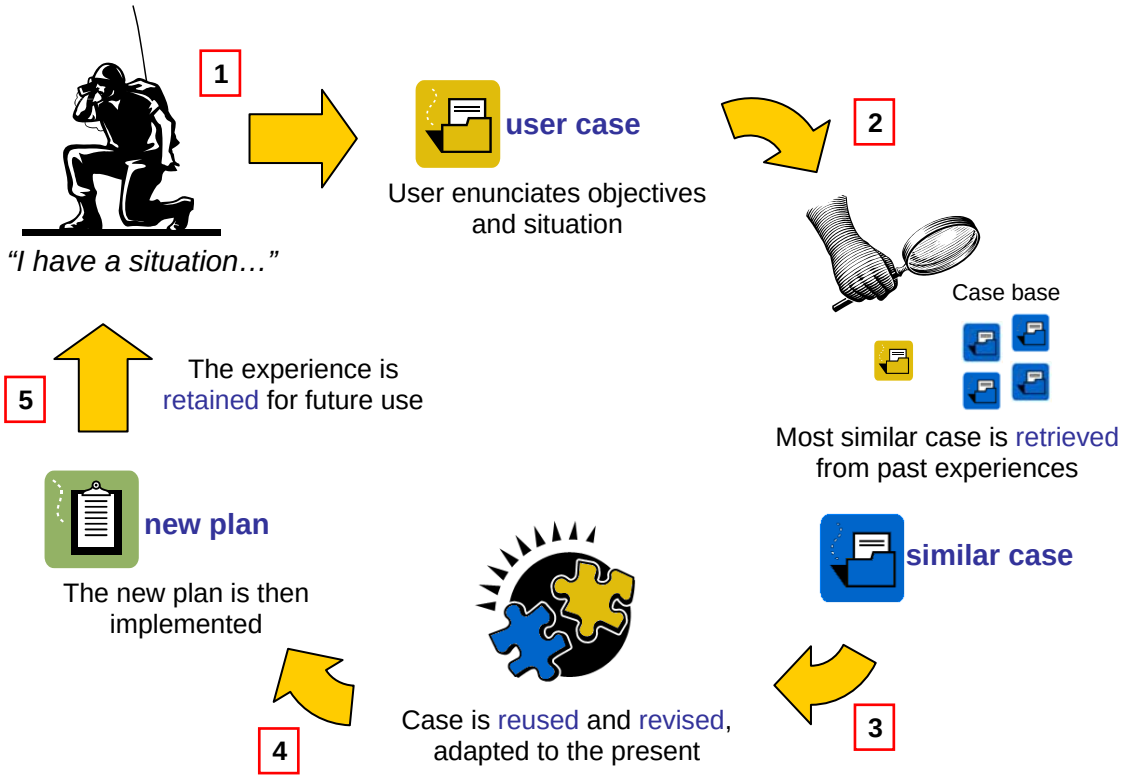

Figure 5 - Case-based Planning

DEEP uses jCOLIBRI (2008), an object-oriented framework in Java for building case-based reasoning (CBR) systems. Figure 5 illustrates how the CBR cycle is applied to case-based planning used in the DEEP architecture. Case-based planning makes use of past experiences to implement new plans and retain their outcomes. That is, "Case-based planning is the idea of planning as remembering" (Hammond, 1990). The planning agent allows the user of the system to input a situation using a user interface. Once the operator feels comfortable with the input, the agent, via the user interface, allows the situation to be forwarded to the blackboard. The situation includes statements about the problem's

objective, locations, actors, resources, and times. This is the primary way the system allows user interaction. While interacting with the user interface, the operator can also view plans on the blackboard and view the case base associated with the planning agent. The case base for each planning agent will be unique.

Once a situation has been placed on the blackboard, the blackboard will broadcast a message notifying all registered systems about the new problem. The listeners in the other planning agents determine what type of object was placed on the blackboard, and react to a new situation by initiating cased-based reasoning for the new problem. See Ford & Carozzoni (2007) for a complete explanation of the CBR process used in DEEP. The CBR process selects the best set of cases from its case base and posts them onto the blackboard as candidate plans. Once the candidate plans are placed on the blackboard, they are processed by the critic agents (discussed in detail below in Section 3.3.4).

Each planning agent is expected to have a unique case base, since each planning agent represents the experience of some entity or group of entities. The case base of an entity can contain experiences of any kind. This variety is readily supported by DEEP's plan representation, CPR, because of its ability to work with planning knowledge at different levels of abstraction.

The interface/planning agent is indeed a multi-faceted entity, providing an interface to the user, an interface to a case base, and an interface to a reasoning engine. These interfaces are important due to tight interaction of these systems. Little processing is done by the planning agent itself, but rather by an external system that it interfaces with (e.g., jColibri). The agent itself is the medium between the reasoning process and the blackboard as well as the human and the blackboard. Now that the plans are on the blackboard and ready for evaluation, it is time to discuss the critic agents.

3.3.4 Critic Agents

The term critic agent is used to describe a number of agents in DEEP that evaluate plans based on a number of criteria. The criteria can be loosely coupled into the following categories: adaptation, critique, and execution. Adaptation critics are plan refiners. Critique agents are evaluators of plans that score based on their own inherent knowledge. Execution critics determine ongoing plans in the DEEP cycle. All of these different types of critics are discussed in depth in the following paragraphs.

3.3.4.1 Adaptation Critic Agents

Adaptation agents in the DEEP architecture are software agents that specialize in further refining a plan based on their particular area of expertise. As explained earlier, the initial plan that is instantiated to the new situation and placed onto the blackboard is a "rough cut" and needs supplementary revision. When the adaptation critic agent receives notification from the blackboard that there is a new instantiated plan on the blackboard, it reviews the plan, makes its changes, and posts a new version of the plan with its adaptations.

There are many possibilities for different adaptation agent specializations. For a proof-of-concept, a Capabilities Adaptation Agent was designed, developed, and integrated into the DEEP architecture. The Capabilities Adaptation Agent's specialization is validating that the actors in the instantiated plan are capable of performing the actions to which they were assigned. In order to accomplish this, it first has to be determined what roles an actor is capable of performing and validate that it is consistent with the action to which it has been assigned. Actors in the DEEP architecture have both default roles as well as specialized roles for the situation that need to be taken into account. If a given actor is not capable of the role that has been assigned, a new actor must be found to replace it. The agent looks in the current situation for similar available actors, where similarity is determined by traversing an actor/role taxonomy and selecting one with a minimal semantic distance (Ford & Carozzoni, 2007). This new, similar actor then replaces the incapable actor in the action. After the agent has adapted this plan using its specialized knowledge, it then posts the updated plan to the blackboard.

The use of Adaptation Agents is also an area where mixed-initiative interaction can once again be brought into the system. Interfaces could be made to allow the plan to be displayed to a user, either as a whole or in a specific way, which would allow the user to apply his or her expertise and adapt part of the plan.

In the process discussed so far, a scenario has been created by specifying objectives, methods and resources that were then placed on the blackboard. Each planning agent has examined its case base to find similar situations and used them to instantiate a new plan based on its past experience. Now that the plans on the blackboard have been instantiated, and refined by multiple Adaptation Agents, they are ready to be scored and criticized by the Critique Agents.

3.3.4.2 Critique Agents

This category of agents can be quite extensive, but the present DEEP system only uses one for demonstration purposes. The particular critique agent implemented is a weather agent, but the future possibilities include political, logistics, ethical, legal and cyber agents, among others.

Scoring agents focus themselves on certain areas of a plan. A weather agent, for instance, would focus on how weather impacts the plan and ignores other areas such as political fallout. A legal agent would not focus on weather, but instead would focus on the legal aspects of the plan. These agents will find and use relevant data and ignore data that is of no concern to them. Critique agents do not change the plan as adaptation agents do; rather, they only analyze how the plan may work in the particular subject area. To have a subject area of expertise, these agents usually wrap or communicate with an outside knowledge source that specializes in that area. The weather agent for example has a weather feed it can communicate with, an understanding of weather rules and the weather capabilities of actors in the plan.

During the DEEP process, critique agents use the adapted plans on the blackboard for evaluation. The agents will use the data they need in the plan to further their processing. For example the weather agent will extract the location data contained in the plan and then use that location data to gain weather information using an external source, such as an RSS (Really Simple Syndication) feed. Once the CPR plan object has been parsed for the needed data, the critique agent will process it. The implementation behind evaluation will be different for each critique agent as will the data required for them out of CPR. It is possible as new critique agents are added CPR will need to evolve to include more information. Once evaluation has finished, these agents use a scoring algorithm to produce a score that is tied to that particular plan, which is posted to the blackboard.

These agents follow a general technical scheme. They implement the Java Agent Development (JADE) Framework, register to and setup listeners to the blackboard, have a knowledge source either internally or externally, have an evaluation implementation, and a scoring algorithm. These agents can also use human sources as their knowledge base allowing for mixed initiative interaction.

3.3.4.3 Execution Selection Critic Agents

Once there are several plans that have been instantiated, adapted, and scored by the various agents in the DEEP system, a final agent is responsible for selecting the top scoring plan(s) and sending them off for execution. This agent is known as the Execution Selection Critic Agent. Its specialization is taking all of the information on the blackboard and using it to evaluate and rank the plans. It then either decides which plan(s) are to be executed by either prompting the user for mixed-initiative input, or selecting one on its own.

The first challenge in developing the Execution Selection Agent was how to know when to notify the agent once the plans on the blackboard have converged. In other words, the problem was how an agent, which by design is not dependent upon other agents, knows when other agents are done accomplishing their tasks. The solution chosen was to utilize the message passing that was discussed in Section 3.2.2. The Interface agents broadcast a message containing information about how many instantiated plans they are posting on the blackboard. When the Adaptation Agents realize they have work to do, they notify the Execution Selection Agent to wait until their work is completed. Similarly, the Critique Agents notify the Execution Selection Agent to wait as well. The Execution Selection Agent receives all these messages and uses them to determine when the appropriate number of adapted plans and scores are placed on the blackboard. The Execution Selection Agent then keeps a count of all the adapted plans and scores that are being placed on the blackboard to determine when the plans have converged.

After the agent realizes that the plans have converged, it then sorts the plans in order of best to worst. In order to sort the plans, it uses the scores associated with each plan to rate them. This functionality does not currently assign a weight to the scores; however, in the future it should.

The Execution Selection Critic Agent also has an interface to pass its plans to simulators to simulate the outcome of certain plans. The idea behind these simulations is that they would be simulation engines that could run quickly, in parallel, and could help give insight about the plans to the selection agent. A more detailed simulation will take place when the top rated plans have been selected. The following section will discuss the more detailed simulation engines and how they will be used.

In our current proof-of-concept demonstration, the Execution Selection Agent sends a message to the Interface Agent containing the top rated plans. The user is then notified that he or she can view the top rated plans and select one to be sent off to the main simulation engine.

3.3.5 Plan Execution

After the Execution Selection agent has decided on the plans to be executed, the DEEP architecture notifies the execution component. While in the long-term DEEP vision execution will be accomplished in the operational world, our current implementation uses simulation to test and evaluate plan execution. The CPR object that has been passed around throughout this cycle contains plan information, but it also contains sections ready to be populated with the plan outcome and events. The plan outcome section is crucial to future reasoning over the plans due to the need to understand what happened while executing that plan in a given context. The simulator has to be able to produce an outcome with the results and how the results were obtained. This is important because it has to be able to document which objectives succeeded or failed and why. This information will populate the events section. An example might be the events leading up to the failure of a supply transport due to enemy interception. The events would describe the specific happenings of that situation, including the enemy planes shooting down the tanker.

Simulation in the DEEP cycle presently involves a randomly generated outcome tied to specific plans. In Section 4.6 we discuss the avenues that are open in the near future for more realistic plan execution.

Once simulated execution of a plan is complete, the user can review the plans on the blackboard. Finally, complete plans containing events and results are assimilated into the case base of the engaged planning agent for future reuse.

4 Results and Discussion

The previous chapters of this report have documented the principled architecture designed for the DEEP project. The explanation of the architecture has also been supplemented by the current prototype proof-of-concept example implementation of several of the DEEP software systems (Section4.1). The initial drive of the DEEP project was to build a research platform that was developed in a modular fashion allowing these future advancements to be easily retrofitted. In this chapter we discuss some of the specific goals that have been established for DEEP as future research areas where the DEEP system shall improve.

4.1 Research Platform Demonstration

The architecture and systems described in the previous chapter have been implemented in a research demonstration prototype. A screenshot of the user interface from the demonstration implementation is shown in Figure 6. This interface provides the user access to his/her case base of experiences, a view of the contents of the system's blackboard, and a mechanism for view evolving plans. This research platform will serve as the basis for all future research in the DEEP program, some of which is described later in this chapter.

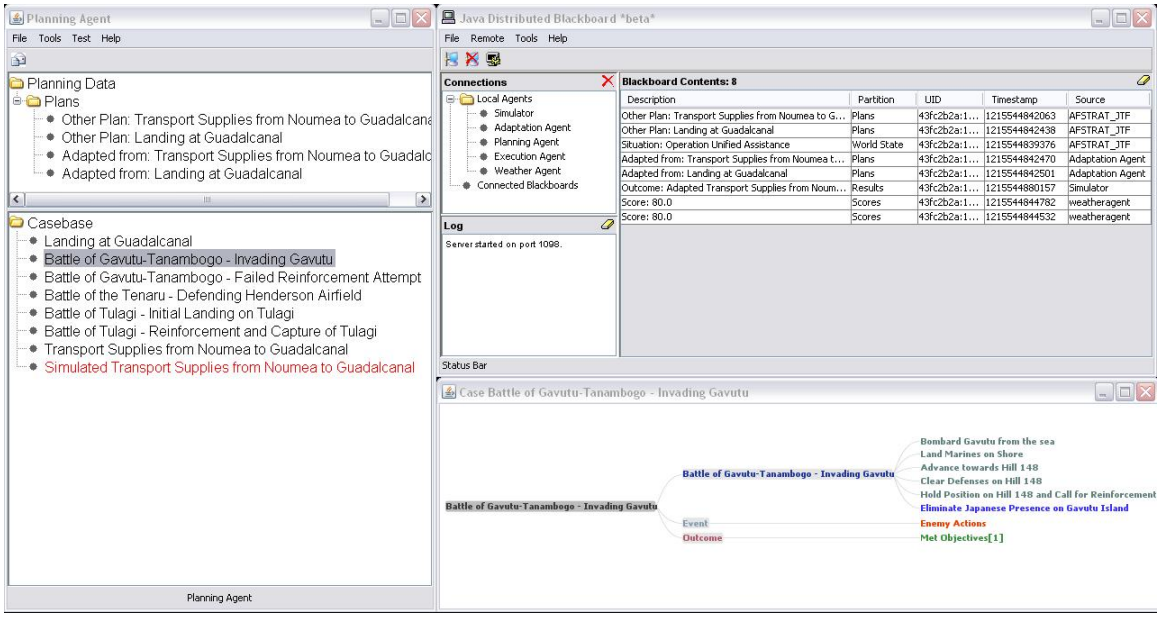

Figure 6 - DEEP Demonstration Prototype

The experience-base of the current DEEP demonstration prototype ("Casebase" in Figure 6) was based on the Battle for Guadalcanal (Aug 42 –Feb 43) and focused on joint aspects of the conflict and not just the management of the air assets. There was strong rationale for using Guadalcanal as the experience-base, including:

- It was an intense, full spectrum combined land/air/sea campaign
- It had the first JFACC-like position (Operation Watchtower)
- It had both traditional and asymmetric aspects
- It included issues addressing cultures and technologies
- It leveraged coalitions and alliances (i.e., Australians)
- It was thoroughly documented and analyzed, a key aspect to facilitate case validation and verification

For the demonstration scenario, a totally different type of mission was chosen: Operation Unified Assistance; the U.S. humanitarian response to the earthquake just north of Sumatra Island and the resulting tsunami. Currently, DEEP demonstrates the power of analogical reasoning in using the historical, military-focused Guadalcanal experience-base to plan for a modern humanitarian relief operation. DEEP will be used to experiment with the capability of analogical reasoning to improved planning speed, plan quality, and plan creativity, with the vision of becoming a truly distributed planning capability in the future.

4.2 Semantic CPR

As discussed in Section3.2.1, the version of CPR that is used in the DEEP system (henceforth called DEEP-CPR to avoid confusion) uses a collection of informal taxonomies to represent plan-related information. While the terms used in these (Pease & Carrico, 1997)taxonomies have meaning to the people who developed them, they are in fact merely collections of symbols that have no explicit meaning to the machine. As such, the interpretation of this information must be programmed into the agents that interpret and manipulate plans represented in DEEP-CPR.

In this section we present an approach to extending DEEP-CPR to leverage semantic technologies. The goal of these extensions will be to allow plans in DEEP to be semantically self-describing. This will allow developers of DEEP system, such as plan analysis agents, to create domain-independent approaches free of hard-coded DEEP-CPR Semantic Extensions

As a first step, we are using RDF (Resource Description Framework) (Brickley & Guha, 2002) as the foundational layer for the semantic extensions to DEEP-CPR. RDF was selected for its simplicity and flexibility to capture and express the semantics within the DEEP planning system. Furthermore, we will attempt to ground the foundational DEEP-CPR concepts with a commonly accepted upper ontology model, such as Suggested Merged Upper Ontology (SUMO) (Pease, Niles, & Li, 2002).

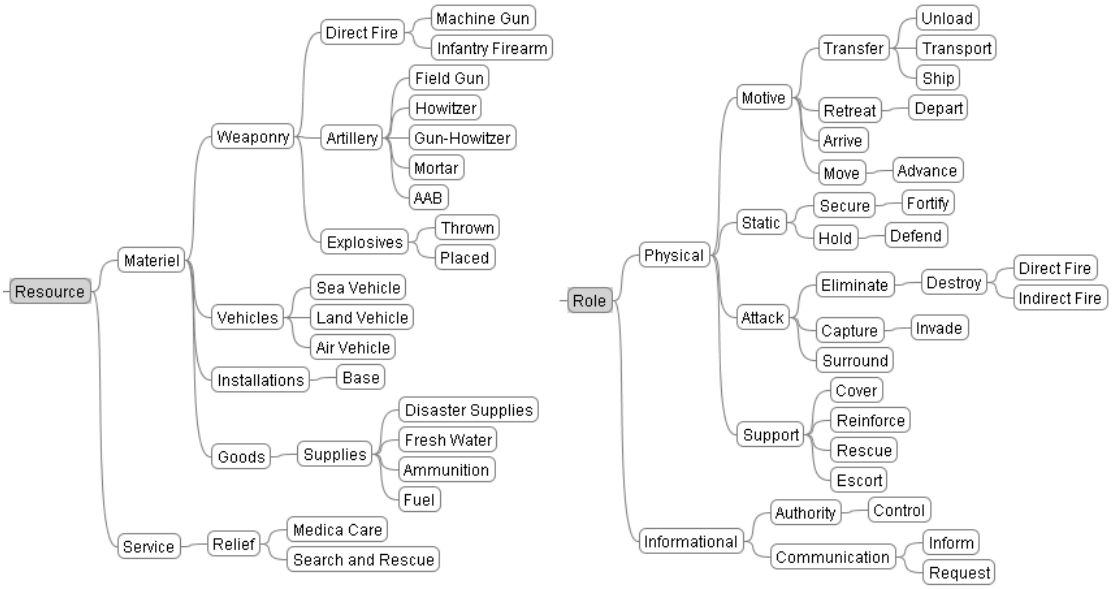

Figure 7 – Resource and Role Taxonomies

For example, the Purpose slot of a Plan object would currently be filled in with a symbol from taxonomy, such as the one shown in Figure 7. Using RDF, it might be represented as shown in Figure 8, which makes statements declaring that Objective is a subclass of, or part of, a Plan, and Purpose is a subclass of an Objective. These statements do not define the meaning of these terms, but rather provide relationships between the terms within a controlled vocabulary.

```
<rdfs:Classrdf:about="&cpr;Objective"
   rdfs:label="Objective">
   <rdfs:subClassOfrdf:resource="&cpr;Plan"/>
</rdfs:Class>
<rdfs:Classrdf:about="&cpr;Purpose"
   rdfs:label="Purpose">
   <rdfs:subClassOfrdf:resource="&cpr;Objective"/>
</rdfs:Class>
```

Figure 8 - RDF Example

Given the difficulty of reading and interpreting RDF statements in XML, these triples can instead be represented in a more simple form:

```
<cpr:Objective  rdfs:subClassOf  cpr:Plan/>
<cpr:Purpose    rdfs:subClassOf  cpr:Objective />
```

4.2.1 URI Reference

To support these extensions, the text-based entries of the DEEP-CPR objects will first be converted to Uniform Resource Identifiers (URIs). A URI is the fundamental building block of semantic technologies. A URI is a character string that encodes a networked resource, providing a structured and stable method for representing concepts and resources in a distributed information space.

Unlike text-based identifiers, which are interpreted by each local system independently, a URI-based infrastructure provides a self-describing data model that ensures agreement on the concepts being used. This is similar to the pass-by-reference model used by modern-programming languages, such as Java, in which information is passed as a pointer to a value rather than a copy of the value. Therefore, when the concept is changed at the source it is not necessary to copy those changes to all instances, since the pointer to that information does not need to change.

HTTP is the most common protocol used to create URIs as it is well suited to resource indexing. In DEEP-CPR, a concept such as the Purpose of a plan could be represented as http://deep.af.mil/namespace/concepts#Purpose. The server containing the reference is *deep.af.mil*, with *concepts* being located in a namespace virtual directory on that server. The *#Purpose* is interpreted as the reference to the Purpose concept within the concepts page.

4.2.2 RDF Metadata Model

Once the DEEP-CPR entries are converted to URIs, they will be encoded into RDF statements. RDF is an important component of semantic technologies being supported by the World Wide Web Consortium (W3C) and the semantic web community. It is a general-purpose language defined by the use of triples to identify and describe networked resources that are represented by URIs. An RDF statement is a triple (S, P, O) in which:

- S is a URI, the subject of the statement
- P is a URI, the predicate of the statement denoting a relationship
- O is either a URI or a literal (plain text) and represents the object or value of the predicate P for subject S

RDF Schema (RDF-S) (Brickley & Guha, 2003)is a vocabulary that introduces additional structure to RDF allowing domain-independent definition of classes, subclasses, and properties. These structural concepts provide a means for declaring relationships between concepts that are generally understood without specific knowledge of the domain. It is this fundamental understanding that allows shared understanding between information spaces without explicit mapping and transformation.

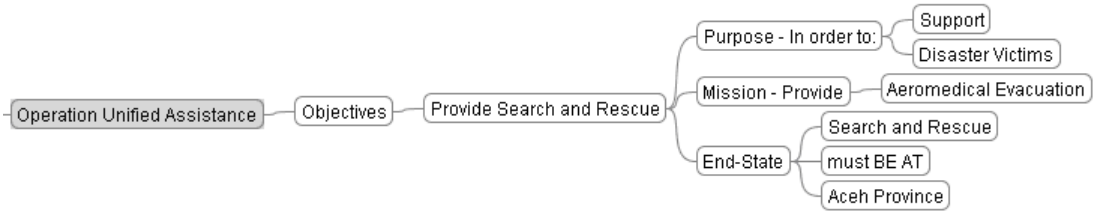

Figure 9 – Objective Decomposition

For example, assume the existence of the following namespaces for *rdfs* and *cpr*, that define the RDF-S and CPR vocabularies. We can then define a case using the abbreviated syntax for a Case called *Operation Unified Assistance*. As in the example shown in Figure, search and rescue assistance will be provided to victims of a disaster in the Aceh region. Concepts that have a numbered suffix, such as `Case2398`, represent the actual internal concept name for unique identification within the system. Since it is possible—even likely, given variations on the same plan—that two cases could have the same name, we cannot rely on the name of the case to distinguish it from another case (Nejdl, 2003). Instead, the name becomes a property of the concept via the *hasName* relationship.

```
[Instance object types]
<Case2398 rdfs:typecpr:Case/>
<Obj8723 rdfs:typecpr:Objective/>
<Pur4423 rdfs:typecpr:Purpose/>
<Role6415 rdfs:typecpr:Role/>
<Entity0123875 rdfs:typecpr:People/>
<Loc23498 rdfs:typecpr:Location/>

[Object properties]
(Alexander, 2004)<Case2398 cpr:hasObjective Obj8723/>
<Obj8723 cpr:hasName "Provide Search and Rescue in Aceh Region"/>
<Obj8723 cpr:hasPurpose Pur4423/>
<Pur4423 cpr:hasName "Support Victims"/>
<Pur4423 cpr:hasRole Role6415/>
<Pur4423 cpr:hasSubject Entity0123875/>
<Role6415 cpr:hasTypecpr:Support/>
<Entity0123875 cpr:hasName "Disaster victims in Aceh region"/>
<Entity0123875 cpr:hasLocation Loc23498/>
<Loc23498 cpr:hasName "Aceh">
```

Each of the concepts used will map to URIs that provide rich textual description defining the meaning of the term. This more explicit definition provides some guarantee that terms used will be based on agreed-upon meanings rather than local interpretation of a symbol. In addition, this self-describing format provides a set of constraints on parts of the plan that will support both comparison and reasoning for the system.

Unlike URIs, RDF statements cannot be stored on web servers as HTML pages. Therefore, it will be necessary to utilize RDF Stores within the DEEP architecture to provide storage of triples. One likely option will be to use an Oracle database, as Oracle will be including support for storage and query of the RDF format (Alexander, 2004)in their next major version. Unlike URIs, RDF statements are more challenging to access in a distributed environment (Nejdl, 2003), although methods such as peer-to-peer (Cai, 2004) indexing have been investigated.

4.2.3 Semantic Technology Benefits and Challenges

The use of semantic technologies as data structures provides significant benefit to both computational and user-based interaction with the information being stored. These benefits include:

- *Expressive.* Since each piece of information is attached to a base hierarchy of structural and functional relationships it can be given more description than a simple string name can provide. Issues such as individual meaning and context can begin to be represented and attached to information to be sure that it is used correctly by the systems that will process it. However, it should be noted the commitment to some first-order logics such as those provided by OWL may also limit the expressivity that can be provided (Pan, 2004). In addition, the notion of a meaning for a given concept will often differ between users, even those within one community of interest. Therefore, it can be a challenge to develop a common vocabulary that represents an agreed upon, shared meaning between multiple users and systems. Additionally, as the meaning of concepts changes over time there need to be established formal methodologies for making changes while still maintaining the integrity of the information being represented.

- *Abstraction*. The hierarchical organization of formal ontologies provides a structure that can be directly interpreted as layers of abstraction. This provides an ideal computational structure for determining the more general classes that subsume a given concept within the case base. This structure could easily be used as a measure of similarity between individual and collections of concepts. Further discussion on the benefits and challenges can be found in (Wiederhold, 1994).

- *Descriptive*. Unlike a flat database schema, the relationships provided by a semantic information space provide more description of the data field than its name and location. Each concept (and accordingly the name used to describe it) can be given any number of properties that provide both supplemental information as well as the relationship it has to other concepts that may be found in the information space. These additional descriptions, while providing new facets of reasoning, also make the information space more difficult to examine and update from a user perspective. Therefore, as this complexity increases attention should be given to ensuring that highly complex descriptions are organized in a way that can be navigated and validated by human users.

- *Longevity*. Formal definitions of information semantics provide a structure that can be represented beyond a specific data source. This allows for concept definitions that extend beyond a current snapshot of the world and enforce a more permanent interpretation. While this does not mean that the meaning of concepts cannot change, it does limit the misunderstandings that can occur based on the use of language alone.

- *Interoperability*. While true context-based interoperability is not yet realized, the structure of semantics provides a more formal basis for promoting predictable data transformation between information spaces. By allowing information spaces to find matching concepts not only by name but also via structural and logical similarities, the likelihood of accurate mapping is increased significantly. However, mapping between these structures is rarely a one-to-one relationship that can be represented in a simple manner. More complex relationships between information spaces require sophisticated functional mappings that are difficult to design, maintain and validate. For example, a 3-wheeled vehicle may be classified as a motorcycle in one model, but have no direct mapping to a model that classifies all vehicles into 2-wheeled and 4-wheeled vehicle categories, even though motorcycle was previously mapped to the 2-wheeled concept.

While most of these benefits are tempered by complications they still provide a more reliable basis for reasoning than can be provided by current flat data standards such as RDBMS and XML. This reasoning support provides significant benefit to the DEEP project by enabling query and inference capabilities that are focused on concepts and relations as opposed to words.

However, there are many challenges that must be overcome before semantic technologies can be widely deployed and maintained by an information infrastructure. This includes:

- *Building ontologies*. Ontologies are commonly built as a manual process in which experts and knowledge engineers brainstorm a concept space and begin to formalize definitions. While this proves effective in smaller domains and information spaces it is often the case that the members of larger communities of interest cannot come to an agreement on the definition of some concepts. Depending on the degree of difference between these views it is sometimes impossible to represent both viewpoints in one ontological structure. Some of the varied methodologies for building ontologies are discussed in (Lopez, 1999).

- *Indexing ontologies*. There are currently no standard methods for indexing and allowing searches over ontological concepts and relationships. While many tools allow for a keyword-based search it is often difficult to know exactly what term is used to describe these constructs. Given the ambiguous nature of language there are often many words that could be used to describe one concept. While significant work has been done regarding mapping ontologies to each other (Maedche, 2002), little work could be found that looked at an infrastructure to provide access to ontologies that have not yet been directly mapped to the local ontology.

- *Ontology versioning*. Just as the world changes constantly, so should ontologies. While it may hold that cars cannot fly today, someday that may change and an ontology describing cars will have to be re-engineered. However, there may be a situation where a system that has a dependency or reliance on the original ontology to provide the same reasoning constructs it provided before. Therefore, it is vital that a reliable standard for publishing and access of ontology versions exist to allow for interaction between information spaces without unexpected change (Klein, 2001).

- *Structure vs. Flexibility*. Since ontologies represent an "upper class" of data, it is important to be wary of the commitments made at the ontological level. Any restrictions placed on a concept will be inherited by any instance that is a type of that concept. If an ontology asserts that all cars are blue then it is not possible for an instance of a car under that ontology to be anything but blue. On the other hand if the ontology contains a concept Car that has no properties or relationship, it is nothing more than a symbol that provides no logical support. Therefore, there is a balance that must be maintained between the amount of structure you impose at the ontology level and the amount of flexibility given to instances of the concepts that have been described.

While working on the current implementation, we found that the landscape of possible specifications was both complex and confusing. For example, it was not initially clear what the differences were between RDF-S terms *subClassOf* and *type*. In addition, RDF-S and OWL share some terms, making it difficult to distinguish between them. Also, we found that the currently available tools are premature and require extensive experience. In fact, RDF generated in one tool will often not open correctly in another tool due to subtle differences between them, even though they are both using the same representation. Further, we believe that RDF encoded in XML format is a misleading representation for human readability even though it is ideal for machine-independent processing. The hierarchical representation reflected by XML encoding is not directly applicable to the structure of the RDF triplets.

However, we believe that the benefits gained should outweigh the shortcomings. Our recommendation to the C2 Community is to invest time and effort into understanding and applying semantic technologies in the form of abstract domain ontologies. More specifically, we suggest using the RDF and RDF-S frameworks as the foundational layer to encode and externalize the semantic concepts within their respective domain. Doing so will enable complex reasoning and portability in the long-term while providing the benefits of lightweight hierarchical reasoning and modularity in the short-term.

4.3 Multi-Case Reconciliation

In DEEP, agents exchange and utilize experiences to solve problems. An interesting challenge that we are exploring is finding a way to use the experiences from several agents to solve a problem. In this section, we discuss the notion of *robust coherence*. Robust coherence involves establishing the truth of experiences in both the collective

context of the problem and the objective world the problem is concerned with. An agent will present a problem, which will be addressed by different agents by allowing each agent to suggest experiences for adaptation. Those experiences will be selected based on their coherence in the collective view of the problem. Also, that set of experiences will be challenged by *critical rationalism* to maintain correspondence with the dynamic world. We will begin with an explanation of *coherence* and conclude this section with a discussion of critical rationalism and how to apply it. Then we will discuss DEEP's Ensemble Case Knowledge (DECK), an algorithm for establishing robust coherence within DEEP. Ford and Lawton (2008) discuss a more detailed outline of this algorithm and its current implementation.

4.3.1 Coherence

Thagard and Verbeurgt (1998) discuss coherence in philosophical, psychological, and computer science terms. Philosophically, coherent knowledge is knowledge that is mutually supportive in an overall context of justification. In simple terms, establishing coherence involves establishing constraints, and satisfying those constraints by sorting elements into either accepted or rejected sets. Solving this constraint satisfaction problem allows a reasoner to determine what information is mutually supportive in an overall system of beliefs. This becomes more complex in the establishment of these constraints, and the many algorithms that are available to solve the resulting constraint satisfaction problem.

This approach requires the establishment of positive constraints and negative constraints. Certain relations can be characterized as coherent (such as explains, associates, or facilitates), while others denote incoherence (such as incompatible, contradictory, or inconsistent). Elements that are related by a coherent relation are positively constrained, while elements related by an incoherent relation are negatively constrained. Fulfilling the constraints established by these relations consists of sorting elements into the appropriate set, accepted or rejected, based on their constraints. Positively constrained elements are either both accepted or both rejected. The two elements that are positively constrained must be sorted into the same set. Negative constraints are satisfied by accepting one or the other element involved. The two elements that are negatively constrained cannot be sorted into the same set. Logically, this is equivalent to treating positive constraints as an AND operation, and negative constraints as an XOR operation. However, this system attempts to maximize constraint satisfaction, which means that not all constraints need to be met, just as many as possible.

Constraints will be satisfied or violated based on a strength value. This strength value indicates how much a constraint contributes to the overall coherence of the system. A constraint satisfaction problem solver determines the most coherent sorting based on these strength values. Constraints may be upheld or violated based on the combination that leads to the greatest coherence. This allows the elements in the system to be accepted or rejected based on the overall coherence of the system.

We can use this notion of coherence to choose a set of experiences that will be acceptable to use in planning or prediction, because they are coherent with the world view established by the group of cooperating agents. In this way, suggested experiences exist in an overall context established by the whole group of agents. Expertise is exchanged collectively, leading to shared understanding of a problem.

While this form of coherence may appear to be useful to establish truth, there is the danger of forming a coherent truth that is circular in nature. That is, forming a set of experiences that mutually support one another as coherent, but do not actually correspond to the outside world and how it really works. Moreover, there could be multiple coherent systems of experiences which lead to alternate world views. Some or none of these equally plausible interpretations could correspond with reality. For this reason, we must be able to establish coherent collections of experience that also correspond with the current reality of a situation.

4.3.2 Critical Rationalism

"It is an old maxim of mine that when you have excluded the impossible, whatever remains, however improbable, must be the truth" (Doyle, 1892)

To address these problems, we turn to another form of reasoning to help inform coherence as an epistemology: a more deductive view of truth that seeks to refute theories based on inconsistency with evidence. Reid and Griffin (2003) discuss this method of reasoning as *critical rationalism*.

Under critical rationalism, theories are postulated and stood to the test of falsification. In other words, theories are considered which are potentially falsifiable, and then compared to a set of observations. If the theory holds up to this set, compared to competing theories, it is considered the least untrue (rather than most true). The measure of this aspect of *truth* is known as *verisimilitude* (Reid & Giffin, 2003). Logically, critical rationalism is based on deduction, rather than induction. This means that verisimilitude measures the degree to which a theory is able to stand up to criticism based on what it deduces should be true.

Reid puts this understanding of knowledge forth as a way to allow military planners to communicate evidence. Under this approach, the utility of information is based on its value in refuting theories, and the importance of doing so. Because an exhaustive search of the complicated world for a complete set of counterexamples is impossible, theories are ranked based on their level of repudiation, not on their level of truth. Communication establishes a reasonable level of verisimilitude for theories, understanding that a complete measure is unattainable.

We can use verisimilitude to better inform a coherent set of experiences by attempting to locate information that refutes some of the aspects of the experiences. By doing this, we establish the degree of false-ness in those experiences for facing a current problem, and avoid the pitfall of blindly applying experience. This is what makes *robust coherence* different from ordinary coherence. Rather than relying on coherence as the only mechanism of justification for beliefs, robust coherence uses critical rationality to establish the "anti-justification" of beliefs.

DECK uses coherence informed by critical rationality to create set of coherent, robust experiences that address a specific problem. The counterexamples for experiences allow us to examine how those experiences' utility is inhibited by facets of the ever-changing world. In this way, critical rationalism can also indicate critical conditions in the world, allowing for the discovery of new goals. In the following section, we will examine the mechanisms DECK employs to accomplish a system of robust coherence.

4.3.3 Planning in DECK

Deliberative coherence is an approach that can be used to determine the appropriate actions and goals in a situation (Thagard & Milligram, 1995). In DECK, plans are formed from previous experience by using actions and goals from recalled experiences as the initial factors in a system of deliberative coherence. This system is populated with facilitation and incompatibility relations that allow the system of coherence to be solved. This means that deliberative coherence does not have to be utilized from scratch, but rather can be accomplished by an ensemble of competent case-based agents.

As different agents suggest experiences from their own case bases, DECK interprets the actions and goals from these experiences in a system of deliberative coherence. This reasoning establishes the positive and negative constraints between these portions of experience, allowing a collective set of actions and goals to emerge as coherent. This collective set can then be adapted and de-conflicted as a cohesive plan.

Throughout this entire process, these experiences are treated like hypotheses by a system of critical rationalism. That is to say, as counterexamples that contradict elements of the experiences enter the system, they weaken the strength of the relations posited by the system of deliberative coherence. Because the strength of the relationships indicates the strength of the constraints in the constraint satisfaction problem, relations with more counterexamples are less likely to be upheld in the constraint satisfaction problem. In this way the solution to the constraint satisfaction problem achieves a system of robust coherence that follows the view of truth established by collective experience and also adheres to the evolving reality of the situation.

4.4 Distributed Database Management System

As discussed in Section 3.3.1, DEEP utilizes a distributed blackboard as the central data structure for knowledge sharing and communication among its various systems. The current implementation (Figure 4) achieves its distributed capabilities by automatically replicating information stored in the primary blackboard module to the remotely connected machines. This is not a robust solution, however, as information may be lost if one of the blackboard modules dies.

To address this shortcoming, we will be replacing the data store component of the blackboard with an Oracle (version 10g/11g) database so that we may leverage its distributed functionality (Figure 10). By utilizing this commercial database system, we can offload many of the issues inherent with a distributed system such as transaction support. We are currently working on integrating a new persistence component that will replace the in-memory data store with a Java interface connecting to an Oracle database. With a connection to an Oracle database management system, not only will we be able to utilize it as a high performance blackboard data store, it will also provide DEEP with a popular integration medium for future integration efforts.

Figure 10 - Distributed Blackboard Architecture using DDBMS

4.5 Formalized Messaging Structure

The messaging system currently employed by DEEP is a simple scheme containing a small number of unique message types, drawn from a taxonomy. One of the reasons to improve the messaging system is to separate the messaging structure from current system so it may be updated and improved as its own component, having minimal impact on the system. The future vision is to leverage research done in *speech act theory* (Cohen & Perrault, 1979) and to formalize a message structure for communication between the various systems of the DEEP system. The JADE Agent Communication Language (ACL) (Bellifemine, 2006)does have some basis in speech act theory, but we need to go beyond relying on the structure of JADE and have a structure that anything in DEEP can adhere to. Instead, we expect to develop a semantically driven language for the system to communicate. The key is that the communicating entities will be involved in a communication where negotiations can happen. By designing a separate and formalized messaging structure in the DEEP project, it will lead to a more robust architecture to be used as a research platform.

4.6 Plan Execution Simulation Options

In Section 3.3.5, DEEP's plan execution simulation was walked through. DEEP is an agent-based system for experienced-based planning and is not developing simulation technology. Instead, DEEP will leverage either COTS simulation packages (i.e. John Tiller's Modern Air Power) or GOTS simulation packages (i.e. Joint Synthetic Forces) to wargame DEEP-developed plans. We intend on only writing just enough "glue code" between DEEP and other simulation systems to hand off developed plans for simulation and in turn to receive the results. The current rudimentary simulation environment currently used in DEEP is a placeholder for future simulation capabilities. For proof-of-concept, the current simulation produces a randomly generated outcome for the plan being simulated. We are considering several simulation alternatives to replace this simple approach.

The first option under consideration is a war-gaming simulator called Modern Air Power (MAP) (Tiller, 2005). Working with Dr. Tiller, our in-house team has developed a programmatic interface to create and control the actions of the various objects in MAP, and to receive feedback on the effects of those actions. The feedback portion of this interface is particularly important, as it is used to make dynamic changes to an executing plan, and to provide outcome information that is included with the plan when it is assimilated back into the case base of experiences. The MAP game engine was designed to support tactical simulation, although it has the ability to model the effects-based operations of a given plan. Because of this design focus, a key challenge of the ongoing MAP interface work has been deriving cases with sufficient information to drive the detailed MAP scenarios.

A second option for the plan execution simulation is an asymmetric agent-based adversarial reasoning simulator called DEEPa, which is being developed by Czech Technical University (CTU) under AFRL/RISB contract (FA8655-07-1-3083). The primary goal of the DEEPa project is the capability to derive opponent behavior patterns.

That is, unlike the MAP simulation, the DEEPa project is using a real-time simulation environment that provides sufficient feedback to interpret *adversarial* actions. Using this information, an adversarial modeling component of DEEPa (Pechoucek, Tozicka, & Rehak, 2006) is able to infer the adversary's intent and dynamically adapt running plans accordingly.

5 Conclusions

In this report we have presented an overview and the progress to date of the Distributed Episodic Exploratory Planning (DEEP) project. The DEEP project was initiated in response to the need to support the key tenets of Network Centric Operations (NCO), namely information sharing, shared situational awareness, and knowledge of commander's intent. To that end, the project's long-term goal is to develop, in-house, a prototype system for distributed, mixed-initiative planning that improves decision-making by applying analogical reasoning over an experience base.

The core of this report documents the successful completion of the project's short-term objective: the development of a "research platform" to further support more aggressive research in the areas of distributed C2 and analogical reasoning, and how to apply the technology to push C2 toward being more Network-Centric. This research platform implements DEEP's high-level system-of-systems architecture comprised of a distributed blackboard, case-based reasoning agents that utilize episodic memories, a system of distributed mixed-initiative planning agents, constructive plan execution simulation, and the ARPI Core Plan Representation (CPR).

Our research platform implementation will serve the DEEP project both as a concept demonstration of how experience-based, distributed mixed-initiative planning can be accomplished in a network-centric environment, as well as an environment for conducting research on the individual system needed to support this NCO vision. We are currently pursuing research in the areas of: semantic extensions to CPR, our plan representation; the use of *robust coherence* to utilize the experiences from several agents to solve a problem; the leveraging of distributed database technology to provide persistent storage for plans and planning information; a speech-act-based messaging formalism for consistent communications among the distributed system; and the use of advanced simulation platforms to improve the fidelity of plan execution and analysis.

The future for DEEP is an exciting one. Although the DEEP project is not focused on building specific tools to hand off to any particular AF-focused customers, we do have a transition path in mind. DEEP will develop and demonstrate technology along with an extensible architecture that supports the long-term vision of distributed planning augmented with analogical reasoning. The transition plans are based on three activities, two of which are discrete events, one is a continuous: the SAB 2009 Information Directorate review, JEFX 2010, and a JFCOM/DARPA collaboration and hand-off.

While not direct transition vehicles, a well-received SAB demonstration and an impressive participation in JEFX 2010 may open doors to a direct transition vehicle. In the current environment, DARPA, which is project-focused, represents a likely transition vehicle. DARPA recently has focused on joining with JFCOM J9 (Joint Concept Development & Experimentation Directorate) and leveraging the Joint Futures Laboratory (JFL). While SAB 2009 and JEFX 2010 are discrete events two years into the future, the JFCOM/DARPA activity can occur as soon as DEEP can demonstrate a

distributed planning demonstration using analogical reasoning. Through this demonstration, DEEP would expect to obtain advocacy from the operational community to enable funding for the future R&D of the complete DEEP system. Alternatively (or even in parallel), once we have a demonstrable prototype DEEP system, we can approach DARPA to develop a program to further develop and advance the state of the art in the key research areas of the DEEP architecture. This avenue would allow us to increase the Technology Readiness Level of the DEEP prototype to facilitate its transition to an operational environment.

6 Bibliography

Alberts, D., & Hayes, E. (2007). *Planning: Complex Endeavors.* Command and Control Research Program.

Alexander, N. L. (2004). *RDF Data Model in Oracle*. (Oracle Corporation) Retrieved from http://download-uk.oracle.com/otndocs/tech/semantic_web/pdf/w3d_rdf_data_model.pdf

Bellifemine, F. (2006). *JADE ADMINISTRATOR'S GUIDE.*

Bellifemine, F. (2006). *JADE PROGRAMMER'S GUIDE.*

Braun, G. (2006). *AFFOR Command and Control Enabling Concept - Change 2.* Internal, USAF/A5XS.

Brickley, D., & Guha, R. (2003, January). *RDF Vocabulary Description Language 1.0: RDF Schema.* (W3C, Producer) Retrieved from http://www.w3.org/TR/2003/WD-rdf-schema-20030123/

Brickley, D., & Guha, R. (2002). *Resource Description Framework (RDF) Model and Syntax Specification*. Retrieved from http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

Cai, M. a. (2004). RDFPeers: A Scalable Distributed RDF Repository based on a Structured Peer-to-Peer Network. *Proceedings of 13th International World Wide Web Conference (WWW2004).* New York.

Caire, G. (2003). *JADE TUTORIAL JADE PROGRAMMING FOR BEGINNERS.*

Cohen, P. R., & Perrault, C. R. (1979). Elements of a Plan-Based Theory of Speech Acts. *Cognitive Science , 3*, 177–212.

Corkill, D. (1990). Blackboard architectures and control applications. *Proceedings 5th IEEE International Symposium on Intelligent Control* . Piscataway, NJ: IEEE.

Corkill, D. (1991, September). Blackboard Systems. *AI Expert , 6* (9), pp. 36-38.

Corkill, D. (2003). Collaborating software: Blackboard and Multi-Agent Systems & the Future. *Proceedings of the International Lisp Conference.* New York.

Doyle, A. C. (1892, Jan-June). The Adventure of the Beryl Coronet. *The Strand Magazine: An Illustrated Monthly , 3*, pp. 511-525.

Ford, A., & Carozzoni, J. (2007). Creating and Capturing Expertise in Mixed-Initiative Planning. *Proceedings of the 12th International Command and Control Research and Technology Symposium.*

Ford, A., & Lawton, J. (2008). Synthesizing Disparate Experiences in Episodic Planning. *Proceedings of the 13th International Command and Control Research and Technology Symposium.*

Hammond, K. (1990). Case-Based Planning: A Framework for Planning from Experience. *Cognitive Science , 14*, 385-443.

Helsinger, A., Thome, M., & Wright, T. (2004). Cougaar: A Scalable, Distributed Multi-Agent Architecture. *Proceeding of the IEEE Sytems, Man and Cybernetics Conference (SMC04).* The Hague: IEEE.

Hughes, C., & Hughes, T. (2003). *Parallel and Distributed Programming Using C++*. Boston: Pearson Education.

jCOLIBRI CBR Framework. (2008, 1 22). Retrieved July 10, 2008 , from GAIA - Group for AI Applications: http://gaia.fdi.ucm.es/projects/jcolibri/jcolibri1/architecture.html

Klein, M. a. (2001). Ontology Versioning for Semantic Web. *Proceedings of 13th Intl' Semantic Web Working Workshop (SWWS'01).* Stanford.

Lopez, F. (1999). Overview of Methodologies for building ontologies. *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI-99) workshop KRR5.* Sweden.
Maedche, A. M. (2002). MAFRA – A MApping FRamework for Distributed Ontologies. *Proceedings of 13th European Conference on Knowledge Engineering and Knowledge Management EKAW-2002.* Madrid, Spain. .
Nejdl, W. S. (2003). Design Issues and Challenges for RDF and schema-based peer-to-peer systems. *ACM SIGMOD Record , 32* (3), 41-46.
Pan, J. a. (2004). *Owl-e: Extending owl with expressive datatype expressions.* Technical Report, Victoria University of Manchester.
Pease, A. (1998). *Core Plan Representation.*
Pease, A., & Carrico, T. (1997). The JTF ATD Core Plan Representation: A Progress Report. *Proceedings of the AAAI Spring Symposium on Ontological Engineering. .* AAAI.
Pease, R. A., Niles, I., & Li, J. (2002). The suggested upper merged ontology: A large ontology for the semantic web and its applications. *AAAI-2002 Workshop on Ontologies and the Semantic Web, Working Notes.*
Pechoucek, M., Tozicka, J., & Rehak, M. (2006). Towards Formal Model of Adversarial Action in Multi-Agent Systems. *Proceedings of the fifth international joint conference on Autonomous Agents and Multiagent Systems (AAMAS).* New York, NY, USA: ACM Press.
Reid, D. J., & Giffin, R. E. (2003). A Woven Web of Guesses, Canto Three: Network Centric Warfare and the Virtuous Revolution. *Proceedings of the 8th International Command and Control Research and Technology Symposium.*
Thagard, P., & Milligram, E. (1995). Inference to the best plan: A coherence theory of decision. In A. Ram, & D. Leake (Ed.), *Goal-driven Learning*, (pp. 439-454).
Thagard, P., & Verbeurgt, K. (1998). Coherence as Constraint Satisfaction. *Cognitive Science, 22 (1)*, pp. 1-24.
Tiller, J. (2005). *Modern Air Power*. Retrieved July 2008, from John Tiller Games: http://home.hiwaay.net/~tiller/modernairpower.htm
Wiederhold, G. (1994). Interoperation, Mediation and Ontologies. *Proceedings of International Symposium on Fifth Generation Computer Systems (FGCS94.* Tokyo, Japan.

7 Symbols, Abbreviations and Acronyms

ACL	Agent Communication Language
AFRL	Air Force Research Laboratory
AOC	Air Operations Center
BBUID	Blackboard Unique Identifier
C2	Command and Control
CBR	Case-Based Reasoning
CPR	Core Plan Representation
CTU	Czech Technical University
DARPA	Defense Advanced Research Projects Agency
DECK	DEEP's Ensemble Case Knowledge
DEEP	Distributed Episodic Exploratory Planning
JADE	Java Agent DEvelopment Framework
JEFX	Joint Expeditionary Force Exercise
JFCOM	Joint Forces Command
JFL	Joint Futures Laboratory
MAP	Modern Air Power
NCO	Network Centric Operation
PMESII	Political, Military, Economics, Social, Infrastructure, and Information
R&D	Research and Development
RDF	Resource Description Framework
RDF-S	RDF Schema
RSS	Really Simple Syndication
SAB	Scientific Advisory Board
SUMO	Suggested Merged Upper Ontology
TRL	Technology Readiness Level
UID	Unique Identifier
URI	Universal Resource Indicator
USAF	United States Air Force